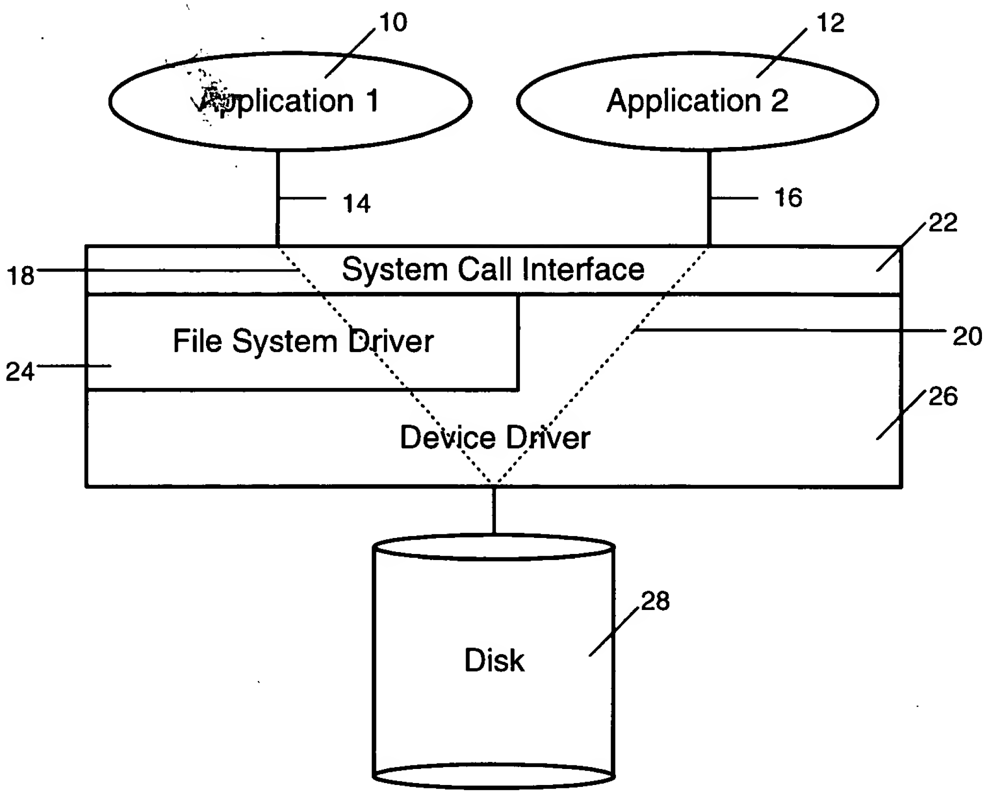
10
12
Application 1
Application 2
14
16
22
18
System Call Interface
File System Driver
20
24
26
Device Driver
28
Disk

Figure 1.

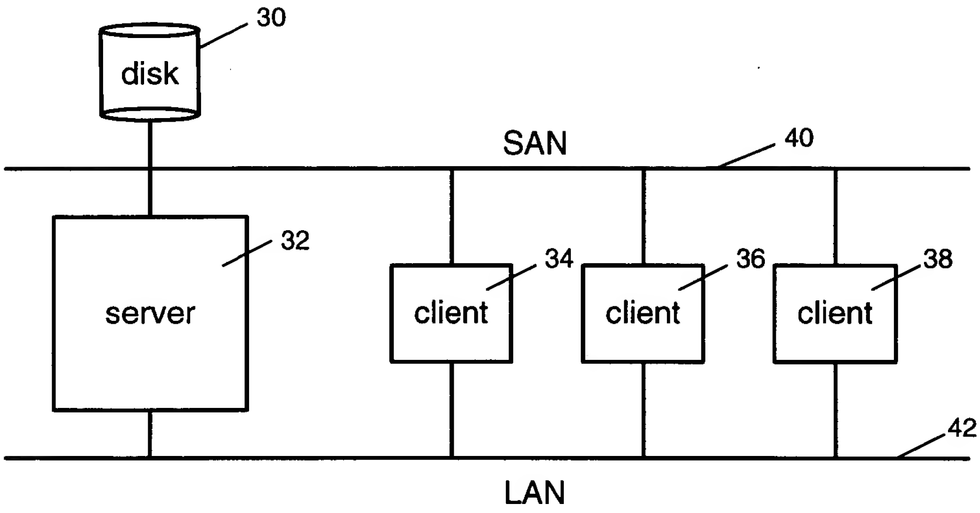

Figure 2.

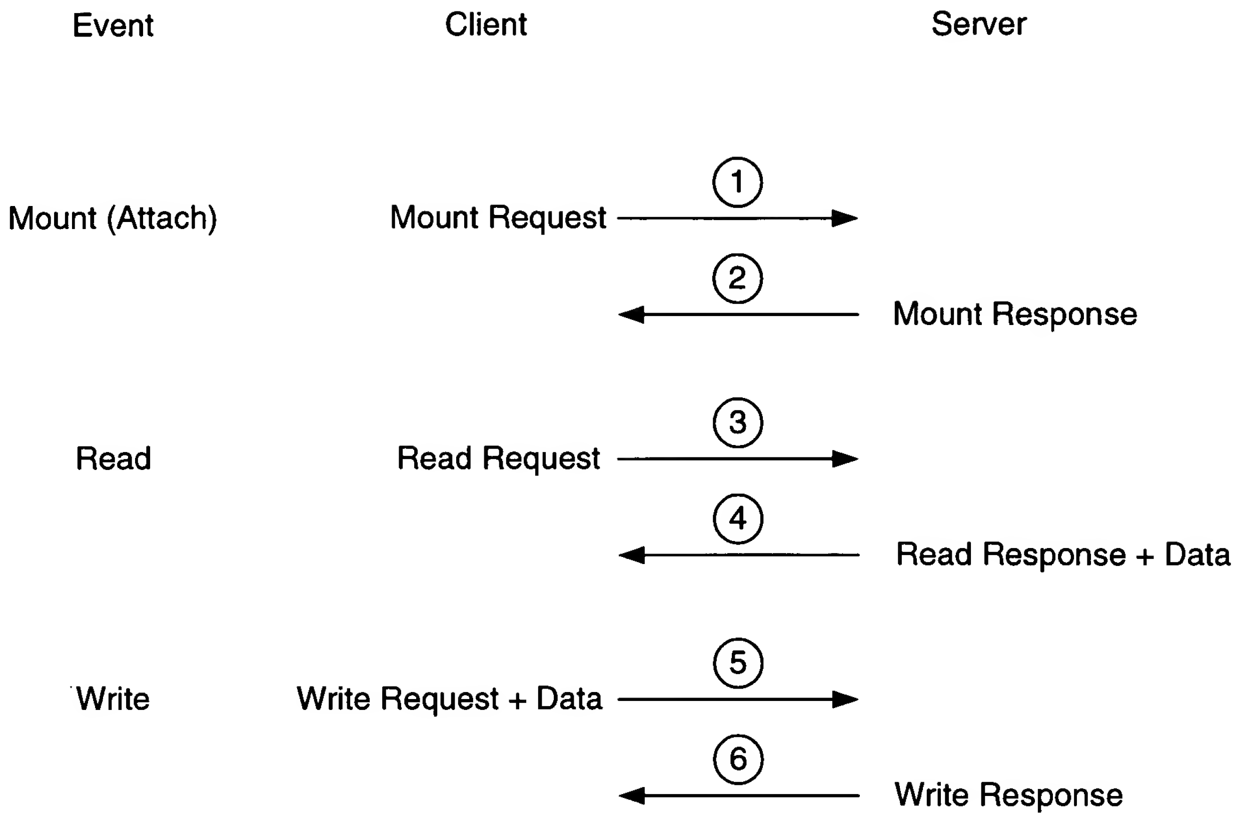
Event
Client
Server
Mount (Attach)
Mount Request
1
2
Mount Response
Read
Read Request
3
4
Read Response + Data
Write
Write Request + Data
5
6
Write Response

Figure 3.

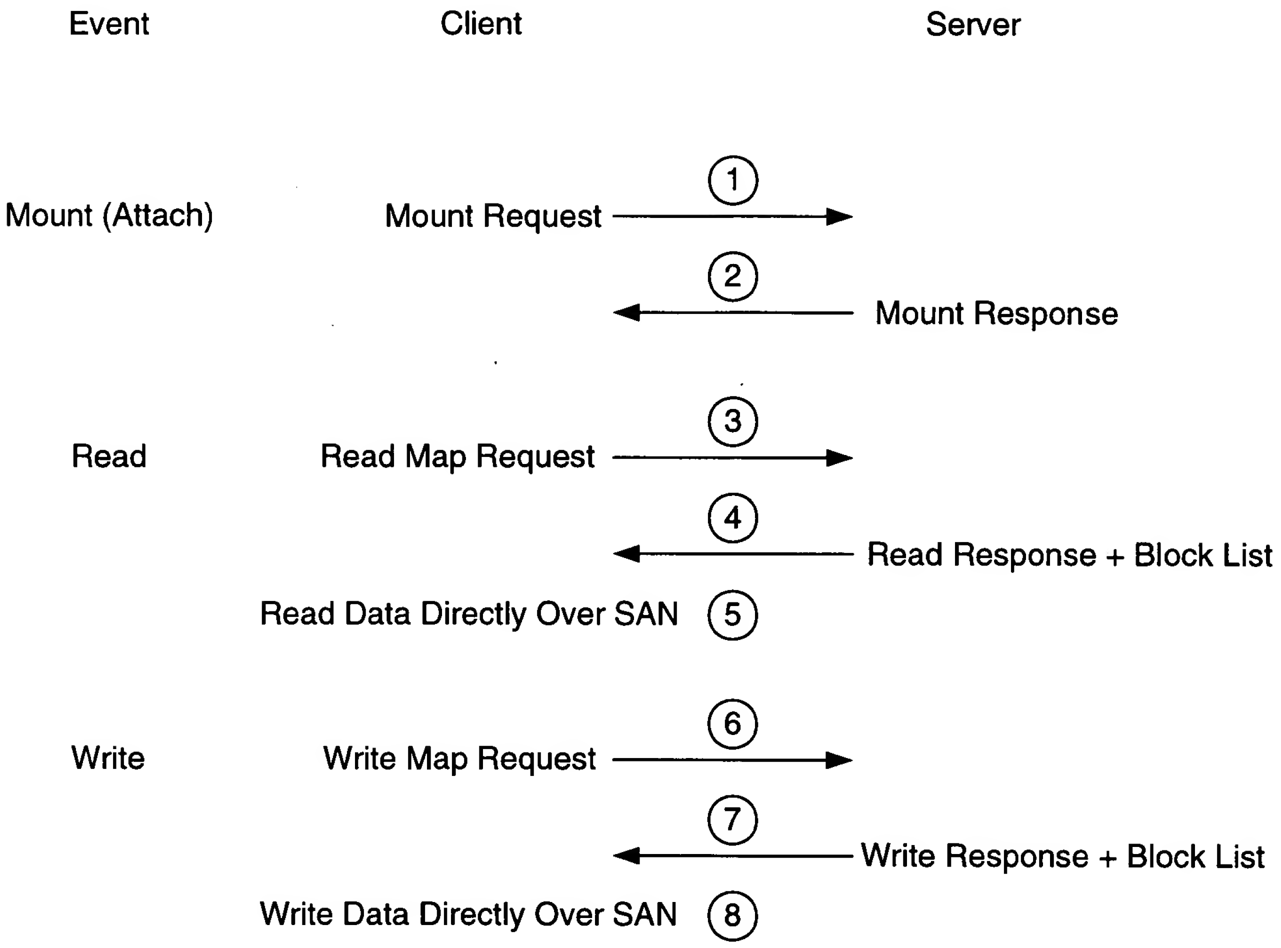
Event
Client
Server
1
Mount (Attach)
Mount Request
2
Mount Response
3
Read
Read Map Request
4
Read Response + Block List
Read Data Directly Over SAN
5
6
Write
Write Map Request
7
Write Response + Block List
Write Data Directly Over SAN
8

Figure 4.

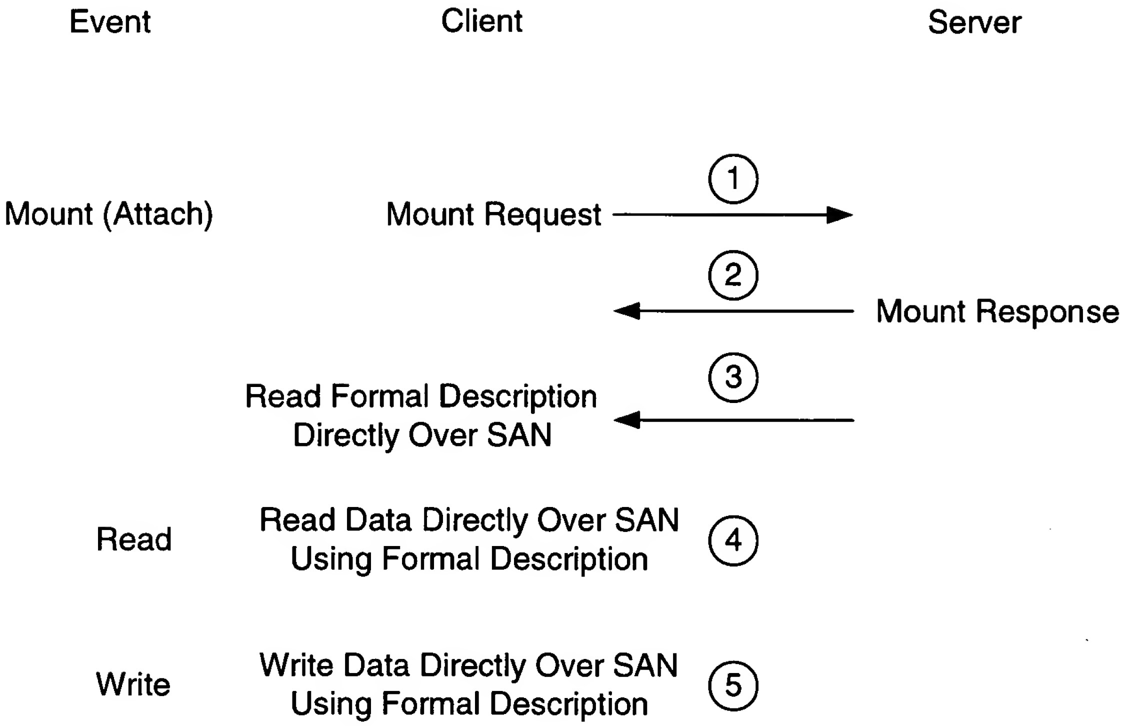
Event
Client
Server
1
Mount (Attach)
Mount Request
2
Mount Response
3
Read Formal Description Directly Over SAN
Read
Read Data Directly Over SAN Using Formal Description
4
Write
Write Data Directly Over SAN Using Formal Description
5

Figure 5.

Feature	Network File System	SNIA Proposal	Self-Describing File System	Clustered File System
Serialization	server	server	server	clients
Disk Block Mapping	server	server	clients	clients
Block Allocation	server	server	server	clients
Disk Access	server	clients	clients	clients

Figure 6.

```
<FS Parameters>
      <CONST NAME="BSIZE"> 1024 </CONST>
      <CONST NAME="INOSZ"> 64 </CONST>
      <CONST NAME="INOPB">  BSIZE / INOSZ </CONST>
      <CONST NAME="ISIZE"> {derived from super block} </CONST>
      <CONST NAME="STARTI"> 2 </CONST>
      <CONST NAME="PSTART"> {physical start of file system} </CONST>
      <CONST NAME="NADDR"> 13 </CONST>
      <CONST NAME="NDADDR"> 10 </CONST>
      <CONST NAME="NIADDR"> 3 </CONST>
      <CONST NAME="IADDRSZ"> 3 </CONST>
      <CONST NAME="IADDROFF"> 12 </CONST>
      <CONST NAME="ISIZEOFF"> 8 </CONST>
      <CONST NAME="BYTEORDER"> 0 </CONST>
      <CONST NAME="NSHIFT"> 8 </CONST>
      <CONST NAME="NINDIR"> BSIZE / 4 </CONST>
      <CONST NAME="NBPSCTR"> 512 </CONST>
</FS Parameters>

<MACRO NAME="LTOPBLK" PARAMS="BN"> BN * (BSIZE / NBPSCTR) </MACRO>

<FUNC NAME="iread">
      <BODY>
            int32
            iread(int16 ino, char *buf)
            {
                  int32 bn;
                  int32 boff;

                  bn = (ino + (2 * INOPB - 1)) / INOPB;
                  boff = (ino + (2 * INOPB - 1)) & (INOPB - 1);
                  PREAD(LTOPBLK(bn), buf, BSIZE);
                  return boff;
            }
      </BODY>
</FUNC>

<FUNC NAME="bmap">
      <BODY>
            int32
            bmap(char *ibuf, int32 off, int32 len, int32 *dbuf)
            {
                  int32 sh;
                  int32 i;
                  int32 j;
                  int32 bn;
                  int32 blim;
                  int32 nblk;
                  int32 *bnp;
                  int32 daddr[NADDR];
                  char ib[BSIZE];
                  char *cp;
                  int32 naddr = 0;

                  nblk = len + (BSIZE - 1) / BSIZE;
                  if (nblk == 0)
                        return 0;
```

Figure 7.

```
                    /*
                     * build an address array, converting from 3-byte
                     * addresses to 4-byte addresses.
                     */
                    cp = ibuf + IADDROFF;
                    for (i = 0, j = 0; i < IADDRSZ; i += IADDRSZ, j++) {
                          if (BYTEORDER == 0)
                                daddr[j] = cp[i]<<16|cp[i+1]<<8|cp[i+2];
                          else
                                daddr[j] = cp[i+2]<<16|cp[i+1]<<8|cp[i];
                    }

                    /*
                     * Fill the dbuf array with the list of block numbers.
                     */
                    while (len > 0) {
                          bn = off / BSIZE;
                          len -= BSIZE;
                          off += BSIZE;
                          if (bn < NADDR - NIADDR) {
                                dbuf[naddr++] = daddr[bn];
                                continue;
                          }
                          bn -= NDADDR;
                          sh = 0;
                          blim = 1;
                          for (j = NIADDR; j > 0; j--) {
                                sh += NSHIFT;
                                blim <<= NSHIFT;
                                if (bn < blim)
                                      break;
                          }
                          if (j == 0)
                                return naddr;
                          ibn = daddr[NADDR-j];
                          if (inb == 0) {
                                dbuf[naddr++] = 0;
                                continue;
                          }
                          for (; j <= 3; j++) {
                                sh -= NSHIFT;
                                PREAD(LTOPBLK(ibn), ib, BSIZE);
                                bnp = (int32 *)ib;
                                i = (bn >> sh) & (NINDIR - 1);
                                if (bnp[i] == 0)
                                      break;
                                ibn = bnp[i];
                          }
                          dbuf[naddr++] = bnp[i];
                    }
                    return naddr;
              }
        </BODY>
  </FUNC>
```

Figure 8.